

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

**Hough Transform Supporting Methods And  
Arrangements**

Inventor:

**Nicholas P. Wilt**

ATTORNEY'S DOCKET NO. MS1-661US

1 **TECHNICAL FIELD**

2 The present invention relates in general to object detection and tracking,  
3 and more particularly to a methods and arrangements for use in detecting visible  
4 features using a Hough Transform.

5 **BACKGROUND**

6 Applications for automatic digital object detection and tracking, image  
7 registration, pattern recognition and computer vision analysis are becoming  
8 increasingly important for providing new classes of services to users based on  
9 assessments of the object's presence, position, trajectory, etc. These assessments  
10 allow advanced and accurate digital analysis (such as pattern recognition, motion  
11 analysis, etc.) of the objects in a scene, for example, objects in a sequence of  
12 images of a video scene. Plural objects define each image and are typically  
13 nebulous collections of pixels, which satisfy some property.

14 These pixels are often the result of some logical operation, such as, e.g.,  
15 filtering, equalization, edge or feature detection, that is applied to the raw input  
16 image data. Hence, objects typically occupy one or more regions within each  
17 image. In a video scene, these objects may change their relative position in  
18 subsequent images or frames depicting the video scene. These objects are  
19 considered moving objects, which form motion within a video scene and can be  
20 automatically detected and tracked with various techniques.

21 The Hough Transform is a well-known computer vision algorithm that can  
22 be implemented to robustly detect a wide variety of features such as lines, circles,  
23 and anything else that can be readily parameterized or otherwise cast in terms of a  
24 discrete popularity algorithm. Unfortunately, the Hough transform tends to be  
25

1 computationally intensive. Many personal computer (PC) microprocessors are  
2 unable to provide the requisite processing capability usually associated with the  
3 Hough Transform. This is more true for video that often requires frame rates of  
4 about twenty frames or more per second, novel user interfaces using computer  
5 vision such as the puppeteering of avatars in online scenarios (e.g., during  
6 massively multiplayer games, virtual teleconferencing, and the like), and feature  
7 extraction for subsequent image processing (e.g., image registration).

8 Thus, to support such capabilities there is a need for improved methods and  
9 arrangements that support the use of Hough transforms, and/or any similar  
10 transform. Preferably, the improved methods and arrangements will be suitable  
11 for implementation in a PC or like device, and/or otherwise realizable in a cost  
12 effective package.

### 13 SUMMARY

14 The above stated needs and other are met by various improved methods and  
15 arrangements that, in accordance with certain aspects of the present invention,  
16 leverage the dedicated hardware of a graphics circuit (e.g., a 3D graphics  
17 accelerator card, etc.) to provide a portion of the data processing associated with  
18 the Hough transform.

19 This may be accomplished, for example, by gathering observations that can  
20 be mapped into a parameter space of a desired feature or features, quantizing the  
21 parameter space of the desired feature(s), and allocating an accumulator and  
22 initializing it to zero for each discrete quantized parameter combination. Then, for  
23 each observation incrementing all of the accumulators that correspond to  
24  
25

1 parameter combinations that might have produced the observation, and finding  
2 maxima in the quantized parameter array.

3 Per the various methods and arrangements provided herein, these basic  
4 processes are selectively performed by a general-purpose processor along with the  
5 dedicated hardware of the graphics circuit. By way of example, in certain  
6 implementations the dedicated graphics hardware provides an alpha-blending  
7 capability that can be leveraged to record votes associated with the Hough  
8 Transform.

### 9 10 **BRIEF DESCRIPTION OF THE DRAWINGS**

11 A more complete understanding of the various methods and arrangements  
12 of the present invention may be had by reference to the following detailed  
13 description when taken in conjunction with the accompanying drawings wherein:

14 Fig. 1. is a block diagram depicting an exemplary computer system that is  
15 suitable for use with the following invention.

16 Fig. 2 is a block diagram of a portion of an exemplary graphics circuit  
17 suitable for use in the computer system of Fig. 1.

18 Fig. 3 is a flowchart depicting an exemplary process for performing a  
19 Hough transform using the circuits and devices in Figs 1-2.

20 Figs 4-8 are data tables illustrating an exemplary Hough transform process  
21 associated with the flowchart in Fig. 3  
22  
23  
24  
25

## DETAILED DESCRIPTION

One of the problems with past implementations of the Hough transform is that the arrays of accumulators needed for Hough transforms have historically needed to be low-resolution because of the data storage expense and execution time requirements for higher resolution arrays. This is because, high-resolution arrays require both more time to manage, and more time to analyze after the algorithm has finished processing the accumulators.

As described below, the various methods and arrangements describe ways to leverage dedicated hardware to solve these problems. As will be shown, it is feasible to use an inexpensive processor that can operate on ordered data to implement the Hough transform. A preliminary/sample implementation of the technique has been implemented on a circa 1998 graphics card. As described in more detail below, in these methods and arrangements, the pixel values within a frame buffer are essentially thought of as accumulators. The methods and arrangements can then use the texture mapping and alpha blending functionality to increment the appropriate accumulators as the Hough algorithm progresses. The following detailed description and accompanying drawings also describe ways to address some limitations in this specific implementation, such as low precision and the inability to use the graphics processor to analyze the output images.

### Computing Environment

Reference is now made to Fig. 1, which is a block diagram of an exemplary computing system 200 suitable for use with the various methods and arrangements in accordance with the present invention.

1 Computing system 200 is, in this example, a personal computer (PC),  
2 however, in other examples computing system may take the form of a special-  
3 purpose device, an appliance, a handheld computing device, a cellular telephone  
4 device, a pager device, etc.

5 As shown, computing system 200 includes a processing unit 221, a system  
6 memory 222, and a system bus 223. System bus 223 links together various system  
7 components including system memory 222 and the processing unit 221. System  
8 bus 223 may be any of several types of bus structures including a memory bus or  
9 memory controller, a peripheral bus, and a local bus using any of a variety of bus  
10 architectures. System memory 222 typically includes read only memory (ROM)  
11 224 and random access memory (RAM) 225. A basic input/output system 226  
12 (BIOS), containing the basic routine that helps to transfer information between  
13 elements within computing system 200, such as during start-up, is stored in ROM  
14 224. Computing system 200 further includes a hard disk drive 227 for reading  
15 from and writing to a hard disk, not shown, a magnetic disk drive 228 for reading  
16 from or writing to a removable magnetic disk 229, and an optical disk drive 30 for  
17 reading from or writing to a removable optical disk 231 such as a CD ROM or  
18 other optical media. Hard disk drive 227, magnetic disk drive 228, and optical  
19 disk drive 230 are connected to system bus 223 by a hard disk drive interface 232,  
20 a magnetic disk drive interface 233, and an optical drive interface 234,  
21 respectively. These drives and their associated computer-readable media provide  
22 nonvolatile storage of computer readable instructions, data structures, computer  
23 programs and other data for computing system 200.

24 A number of computer programs may be stored on the hard disk, magnetic  
25 disk 229, optical disk 231, ROM 224 or RAM 225, including an operating system

1 235, one or more application programs 236, other programs 237, and program data  
2 238.

3 A user may enter commands and information into computing system 200  
4 through various input devices such as a keyboard 240 and pointing device 242  
5 (such as a mouse). Of particular significance to the present invention, a camera  
6 255 (such as a digital/electronic still or video camera, or film/photographic  
7 scanner) capable of capturing a sequence of images 256 can also be included as an  
8 input device to computing system 200. The images 256 are input into computing  
9 system 200 via an appropriate camera interface 257. This interface 257 is  
10 connected to the system bus 223, thereby allowing image data to be stored in  
11 RAM 225, or one of the other data storage devices. It is noted, however, that  
12 image data, such as COM data, can be input into computing system 200 from any  
13 of the aforementioned computer-readable media as well, without requiring the use  
14 of the camera 255.

15 As shown, a monitor 247 or other type of display device is also connected  
16 to the system bus 223 via an interface, such as a video adapter 245. In addition to  
17 the monitor, computing system 200 may also include other peripheral output  
18 devices (not shown), such as speakers, printers, etc.

19 Computing system 200 may operate in a networked environment using  
20 logical connections to one or more remote computers, such as a remote computer  
21 249. Remote computer 249 may be another personal computer, a server, a router,  
22 a network PC, a peer device or other common network node, and typically  
23 includes many or all of the elements described above relative to computing system  
24 200, although only a memory storage device 250 has been illustrated in Fig. 2.  
25

1 The logical connections depicted in Fig. 2 include a local area network  
2 (LAN) 251 and a wide area network (WAN) 252. Such networking environments  
3 are commonplace in offices, enterprise-wide computer networks, Intranets and the  
4 Internet.

5 When used in a LAN networking environment, computing system 200 is  
6 connected to the local network 251 through a network interface or adapter 253.  
7 When used in a WAN networking environment, computing system 200 typically  
8 includes a modem 254 or other means for establishing communications over the  
9 wide area network 252, such as the Internet. Modem 254, which may be internal  
10 or external, is connected to system bus 223 via the serial port interface 246.

11 In a networked environment, computer programs depicted relative to the  
12 computing system 200, or portions thereof, may be stored in the remote memory  
13 storage device. It will be appreciated that the network connections shown are  
14 exemplary and other means of establishing a communications link between the  
15 computers may be used.

### 16 17 **Exemplary Graphics Hardware**

18 Fig. 2 depicts a portion 300 of an exemplary video adapter card 245, having  
19 a graphics processor 302 that is operatively coupled to interface 304 and local  
20 memory 306. As illustratively shown, graphics processor 302 includes a transform  
21 mechanism 310 that is capable of performing one or more transform functions  
22 using information provided in local memory 306. In this manner, graphics  
23 processor 302 is able to process graphics information associated with a frame  
24 buffer 308. Interface 304 provides an interface for other circuits to graphics  
25 processor 302 and local memory 306. As such, processing unit 221 is able to



1 provide inputs to graphics processor 302 and access local memory 306, or  
2 otherwise receive outputs there from, via interface 304.

3 Most PCs have a graphics processor with significant processing  
4 capabilities. For example, it is not uncommon for a PC to have a graphics  
5 processor that can process a couple hundred million values in an ordered array  
6 each second. One example is NVIDIA's Quadro2 MXR, which can process 25  
7 million triangles per second and 400 million pixels per second. These powerful  
8 graphics processors are typically coupled to a significantly large and fast local  
9 memory, e.g., 32 Mbytes, 64Mbytes, 128Mbytes, of SDRAM.

### 10 11 **Exemplary Applications of the Hough Transform**

12 The Hough transform employs a well-known algorithm that can robustly  
13 detect features by constructing and analyzing a data structure using a number of  
14 distinct observations.

15 The feature to be detected must be parameterizable, such as a line that can  
16 be described as the parameters  $(\theta, \rho)$  in the equation  $\rho = x \cos \theta + y \sin \theta$ . In  
17 addition, it is necessary to be able to identify all parameter sets corresponding to  
18 the features in which a given observation could participate.

19 Once these criteria have been fulfilled, the feature detection is implemented  
20 using a bin counting algorithm. The parameters are quantized and for each  
21 discrete combination of parameters, an accumulator is allocated and initialized to  
22 zero. Then, for each observation, the accumulators corresponding to every set of  
23 parameters that satisfies the equation for that observation are selectively  
24 incremented.

1 Attention is now drawn to Fig. 3. As depicted in Fig. 3, an exemplary  
2 algorithm proceeds as follows. In initialization step 402 a portion of the parameter  
3 space that may contain a desired feature(s) is quantized and, for each discrete  
4 quantized parameter combination an accumulator is allocated and set to zero (0).

5 In preprocessing step 404, observations that can be mapped into the  
6 parameter space of the desired feature(s) are gathered. Next, in accumulation step  
7 406, for each observation, all of the accumulators that correspond to parameter  
8 combinations that might have produced the observation are incremented.

9 In post processing step 408, the maxima in the quantized parameter array is  
10 found and the parameter combinations whose elements are maximum correspond  
11 to features detected by the set of observations gathered in step 402.

12 An example of basic process 400 in Fig. 3 will now be described. The  
13 following example is for purposes of illustration only. Many of the details of this  
14 example, including the type of feature being detected, the input format, the type of  
15 observation used to detect the feature, the edge detection operator, the method of  
16 gathering and selecting the observations that will participate in the accumulation  
17 process, and the resolution of the parameter quantization may vary from what is  
18 described below in practice.

19 This example will illustrate edge detection in an image using the Hough  
20 algorithm as provided in Fig. 3. The input is an image of grayscale pixel values;  
21 the output is the line equation of the detected edge, using the parametric equation  
22  $\rho = x \cos \theta + y \sin \theta$  for the line. The observations used to detect the edge are  
23 locations  $(x_0, y_0)$  and differences  $(\Delta x, \Delta y)$  reported by a convolution-based gradient  
24 detection algorithm. The locations and differences may be mapped into the  
25 parameter space  $(\theta, \rho)$  using the following equation:

$$\theta = \tan^{-1} \frac{\Delta y}{\Delta x}$$

$$\rho = x_0 \cos \theta + y_0 \sin \theta$$

For each observation, the accumulator corresponding to the  $(\theta, \rho)$  pair is incremented. After each observation has been recorded, the accumulators are searched for maxima. The accumulators with large values correspond to lines detected by the observations.

An exemplary 10x10 matrix 500 of pixel values that contain an edge is depicted in Fig 4.

Next, as depicted in Fig, 5, matrix 500 is transformed for edge detection using horizontal and vertical 3x3 Sobel operators, 502 and 504, respectively. This results in a corresponding 8x8 horizontal matrix 506 and 8x8 vertical matrix 508, wherein the resulting Sobel value operator values near the edge are larger.

Next, as depicted in the Fig. 6, an 8x8 matrix 510 is determined by taking the magnitude  $\sqrt{H^2 + V^2}$  of the Sobel operators.

Table 520 in Fig. 6, lists the observations, i.e., locations where the Sobel operator reported a magnitude greater than zero.

Next, as depicted in Fig. 8, in an array 530 of accumulators (i.e., a voting buffer) corresponding to  $(\theta, \rho)$  pairs is allocated and initialized to 0. For purposes of this example,  $\theta$  is quantized in the range  $[0, 45]$  in increments of 5 degrees and  $\rho$  is quantized from 2 to 8 in half-point increments. The resulting array 530 is depicted with zeros left blank. As can be seen, the maximum value in array 530 is 8, wherein  $25 \leq \theta \leq 30$  and  $4 \leq \rho \leq 4.5$ . Consequently, this result gives a strong and accurate indication of where the edge is in the image (see Fig, 4).

The true power of the Hough transform comes to light when the observations are noisy, and when limited information is available about them.

1 If the observations are noisy, spurious observations have a negligible  
2 impact on the overall bin counting. In the above example, spurious observations  
3 that caused up to 7 votes to be tallied in an incorrect accumulator would still yield  
4 a correct result.

5 If limited information is available about the observations, the bins  
6 corresponding to all possible features that could explain the observation could be  
7 incremented. Using the previous example, if the algorithm reports only locations  
8 and not angles, then the Hough algorithm can increment the accumulators  
9 corresponding to *all lines* that intersect a given location. Maxima in the resulting  
10 accumulator array 530 would still identify lines in the image robustly.

11 The Hough algorithm can also detect features that are not lines. Circles, for  
12 example, may be parameterized as  $(x_0, y_0, \rho)$  in the equation  
13  $(x - x_0)^2 + (y - y_0)^2 = \rho^2$  where  $(x_0, y_0)$  are the center of the circle and  $\rho$  is the  
14 radius. A 3D array of accumulators may be used to search for circles with different  
15 radii, or a 2D array of accumulators could be used if the radius is known.

16 The Hough algorithm can also be applied to robust template matching  
17 (“generalized Hough”). One possible method for doing so is to pick a reference  
18 point on the template object and compute a series of gradients along the boundary.  
19 For each gradient, the angle and distance from the reference point is stored. The  
20 array of accumulators in accumulation step 406 of the algorithm corresponds to  
21 the possible locations of the object being searched for. The observations in step  
22 404 give gradients and locations that can be used in conjunction with the angles  
23 and distances of the gradients to ‘vote’ for the possible locations of the reference  
24 point of the template. When the voting is complete, the accumulator with the  
25 highest value corresponds to the location of the feature.

1 Certain advantages of the invention extend to the above-described  
2 implementation of generalized Hough, and improvements of the algorithm such as  
3 taking the gradient strength into account when determining which accumulators to  
4 increment.

5 The size of a naïvely allocated array of accumulators increases  
6 exponentially in the number of parameters. One way to reduce the number of  
7 elements in this array is to allocate them lazily, deferring the allocation of the  
8 memory needed for a given accumulator until that accumulator must be  
9 incremented. This approach makes sense if the observations map into a sparse set  
10 of the accumulators that encompass the quantized parameter space. This  
11 technique may be applied to the Invention by lazily allocating arrays for the  
12 algorithm to operate on.

13 Another extension of the naïve Hough algorithm described above is to  
14 apply a more sophisticated operation to the accumulator than a simple increment.  
15 Here, for example, one could take the observation error into account when  
16 incrementing the accumulator. Hence, a tent function is added to the accumulators  
17 that correspond to the features that could produce the observation. Larger values  
18 are applied to accumulators that are more likely to have produced the observation;  
19 the falloff of the tent function should reflect the error characteristics of the  
20 observation in question.

21 To make this work, it is best to arrange the array of accumulator so the  
22 accumulators for related parameters may be treated as contiguous. Discontinuities  
23 such as the  $-\pi/\pi$  discontinuity in angular measurements can be worked around  
24 with addressing calculations.  
25

1 With the previous sections in mind, it has been found that a 3D accelerator  
2 may be used to accelerate the incrementing of the appropriate accumulators when  
3 running the Hough algorithm. An overview of the algorithm as implemented on a  
4 3D accelerator is as follows.

5 Preprocessing step 402: gather observations that can be mapped into the  
6 parameter space of the desired feature(s).

7 This step may be performed with the aid of graphics processor 302,  
8 provided it supports the needed operations. For example, if graphics processor 302  
9 can perform convolution and report the results back to processor 221 (or report  
10 locations and convolution values where the operator exceeds a specified  
11 threshold), then processor 221 would not have to perform any pixel processing to  
12 gather the observations.

13 Initialization step 404: quantize the parameter space of the desired  
14 feature(s) and, for each discrete quantized parameter combination, allocate an  
15 accumulator and initialize it to 0.

16 This step entails allocating frame buffer 308 to contain the accumulators.  
17 Each surface contains a 2D slice of quantized parameter space; if 3 or more  
18 dimensions are being accumulated, the third and higher dimensions can be  
19 quantized into an array of discrete frame buffer surfaces.

20 Accumulation step 406: for each observation, increment all of the  
21 accumulators that correspond to parameter combinations that might have produced  
22 the observation.

23 This step may be implemented using alpha blended triangles. The alpha  
24 blending stage replaces each output pixel with  $Src * \alpha_{rcs} + Dest * \alpha_{dest}$ , where  $Src$  is  
25 the source fragment;  $Dest$  is the pixel in the frame buffer location; and  $\alpha_s$  and

1  $\alpha_d$  are application-specified alpha factors that may originate from the fragment,  
2 the interpolants, or the frame buffer.

3 For simple addition,  $\alpha_s$  and  $\alpha_d$  may be set to 1. The vertices of the  
4 triangles may be used to control the intensity of the value being added. A texture  
5 may be used as a lookup table to add an arbitrarily complex function, such as a  
6 Gaussian curve, or the like, approximating the error of the observation, over the  
7 accumulators in the frame buffer 308.

8 Numerous triangles can describe an arbitrarily complex function to apply to  
9 the arrays of accumulators. For arrays of accumulators of 3 or more dimensions,  
10 triangles could be used to apply the tent function over a 3D volume of  
11 accumulators in the array of surfaces.

12 For large numbers of observations, precision in frame buffer 308 may  
13 become an issue. The saturated arithmetic performed when alpha blending serves  
14 us in good stead when this happens – values that are already maximum stay at the  
15 maximum. If contiguous regions become saturated in the frame buffer arrays, the  
16 post processing phase must look for the centroids of regions of maximum values.

17 Post processing step 408: find maxima in the quantized parameter array.  
18 The parameter combinations whose elements are maximum correspond to features  
19 detected by the set of observations gathered in Step 1.

20 As with Step 402, step 408 may be accelerated if the graphics processor  
21 contains logic that can aid with feature detection of the pixel data. For example,  
22 hardware-accelerated histogram computation could be used to implement a  
23 hardware-accelerated  $O(\lg N)$  search for maxima in the output images.

24 Moreover, graphics hardware that can operate on multiple channels  
25 simultaneously and in parallel enables parallel processing of parameter

1 accumulation. For example, each element in the accumulator array could contain  
2 some number of elements that could be incremented by different amounts  
3 (including zero) on an individual basis. A single array could then contain  
4 accumulators for different types of features, or a single frame buffer could contain  
5 multiple levels of a quantized 3D parameterization. More specifically, with  
6 respect to 3D graphics hardware, using the color channels in the frame buffer  
7 allows different accumulators to coexist in the same frame buffer. The vertex  
8 colors and modulation and texture blending hardware interact to control how the  
9 individual components are incremented.

10 The Hough transform techniques taught herein also benefit from higher  
11 precision in the accumulators. As the number of observations grows, the  
12 probability increases that elements in the accumulator arrays will be saturated,  
13 thereby reducing the accuracy of the detection. More specifically, with respect to  
14 3D graphics hardware, increasing the precision of color components beyond 8 bits  
15 per pixel would result in a corresponding and advantageous increase in precision  
16 of the detection.

17 Furthermore still, for hardware that can treat the accumulators as inputs, the  
18 technique may be applied recursively to the output of a previous pass of the  
19 algorithm. Obviously the features detected by a later pass need not be the same or  
20 even remotely related to the features detected earlier.

21 Although some preferred implementations of various methods and  
22 arrangements have been illustrated in the accompanying Drawings and described  
23 in the foregoing Description, it will be understood that the invention is not limited  
24 to the exemplary implementations disclosed, but is capable of numerous  
25



rearrangements, modifications and substitutions as set forth and defined by the following claims.

007654855-000004